Case Study: Modelling Industrial Dryer Temperature

Arun K. Tangirala

17/11/2017

Background

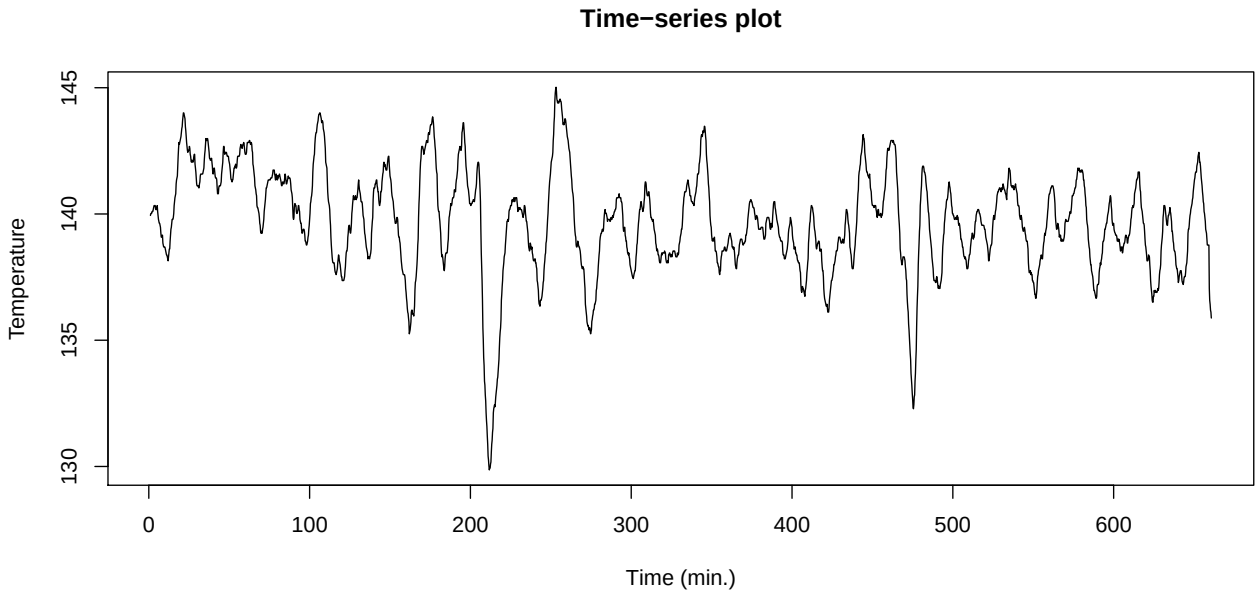
This is a case study concerning time-series modelling of the temperature of an industrial dryer. Data set contains 2640 observations obtained at a sampling interval of 15 seconds (4 observations in a minute).

Through this case study, we shall learn how to develop a time-series model following a step-by-step procedure.

Preliminary analysis

We shall first load the data, create the time-series object (if the given data is not one) and perform a preliminary analysis including visualization of the series (this is important).

```
load("../datasets/tempdryer.Rdata")
# Convert to a time-series object
vk <- ts(tempdryer, frequency = 4, deltat = 1)
# Plot the series
plot(vk, main = "Time-series plot", xlab = "Time (min.)", ylab = "Temperature")
```



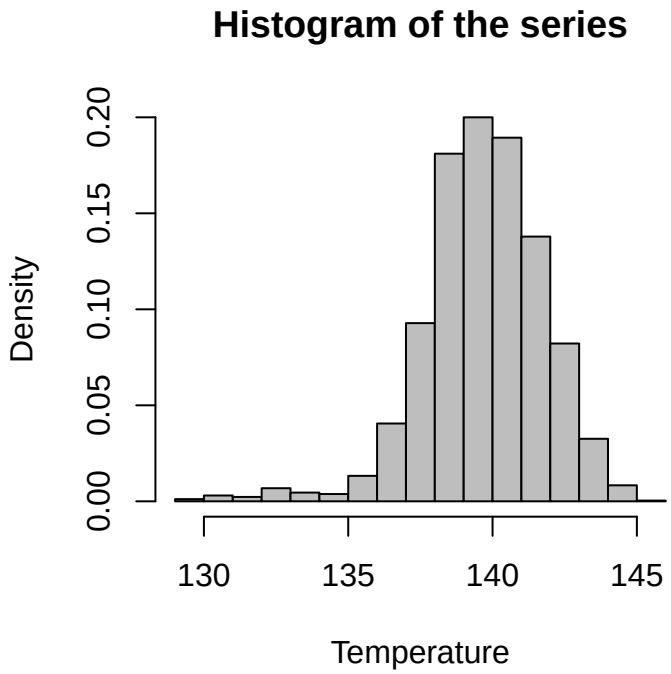

```
# Obtain the statistical summary
summary(vk)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   129.9   138.5   139.8   139.7   141.0   145.0
```

It is clear from the plot as well as the statistic that the series (call it $v[k]$) has (naturally) a non-zero mean.

Linear time-series models assume that the random process is jointly Gaussian (for optimal predictions). While it is very difficult to verify this assumption rigorously, a rudimentary check can be performed by examining the histogram.

```r
hist(vk, probability = T, main = "Histogram of the series", col = "gray", xlab = "Temperature")
```



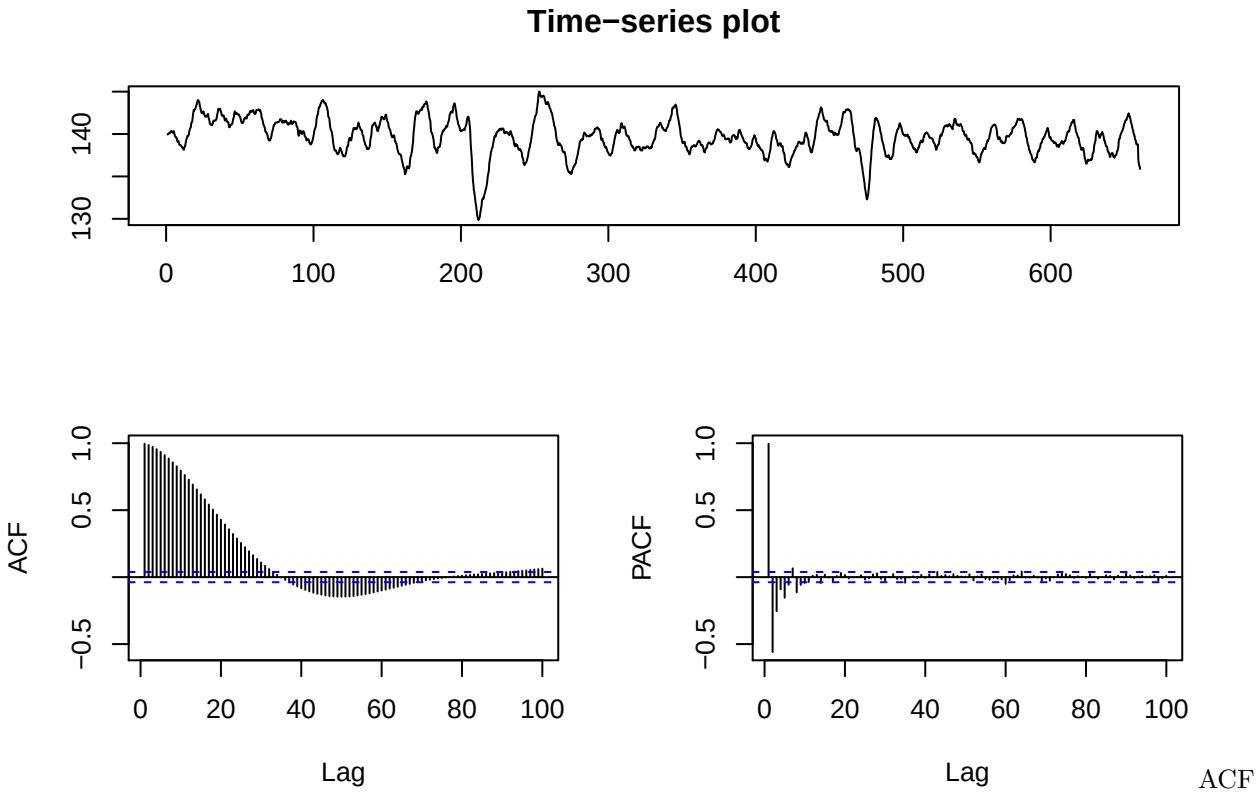
The histogram indicates that the assumption of Gaussianity mildly holds with a stretched out one-sided tail. For our modelling purposes, we shall ignore this deviation from our ideal expectations and proceed with our usual step of examining the ACF and stationarities.

Non-parametric analysis

From a visual analysis, there appears to be some evidence that the series exhibits non-stationarities, especially the variance type. Variance non-stationarities can be either due to integrating effects or due to heteroskedasticity. The first step is to examine the ACF and PACF.

ACF and PACF

```r
tsdisplay(vk, lag.max = 100, main = "Time-series plot", points = F)
```

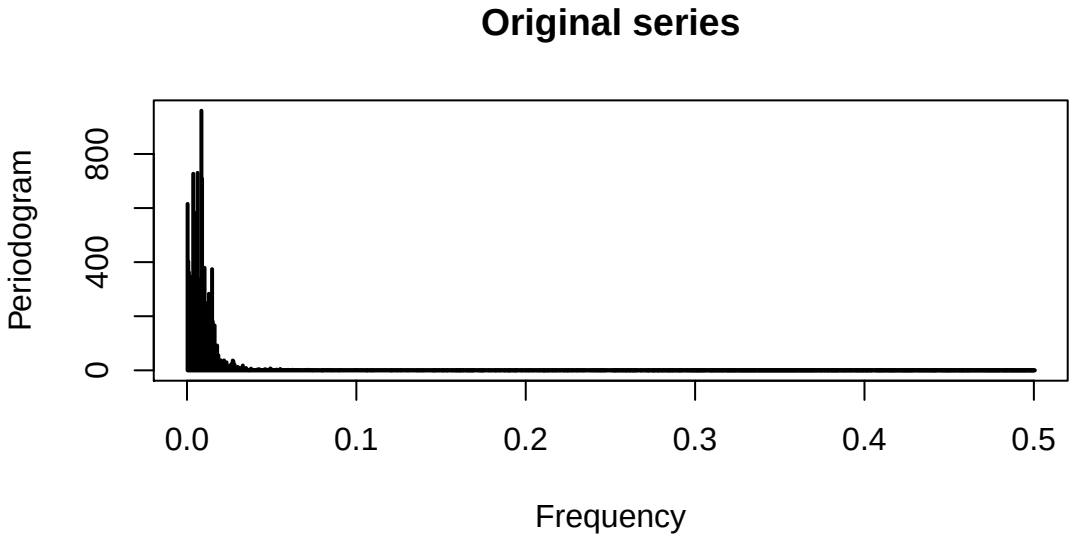


ACF plot shows a slowly decaying correlation, suggesting the possibility of integrating effects coupled with some stationary behaviour and periodicity (or seasonality). We can conduct a unit root test to confirm the presence of integrating effects. Periodicities (seasonality) can be tested by first examining the periodogram followed by a classical (seasonal + trend + level) decomposition, if necessary.

Periodogram

We shall use the `periodogram` routine from the `TSA` library to generate the periodogram (alternatively one could use the `spectrum` from the `stats` package, but with appropriate options)

```
periodogram(vk, xlab = "Frequency", ylab = "Periodogram", main = "Original series")
```



From the periodogram plot it is clear that most of the power in the process is in the low frequency regime with heavy contributions from near-zero frequencies (once again indicative of the presence of integrating effects). The periodicities, if any, can be highlighted further if we are able to eliminate the extremely low-frequency components (perhaps due to integrating effects).

Testing for integrating effects

To confirm the presence of integrating effects, we turn to one of the popular **unit root** tests, known as the KPSS test. In `R`, the `kpss.test` routine in the "tseries" package conducts this test for us.

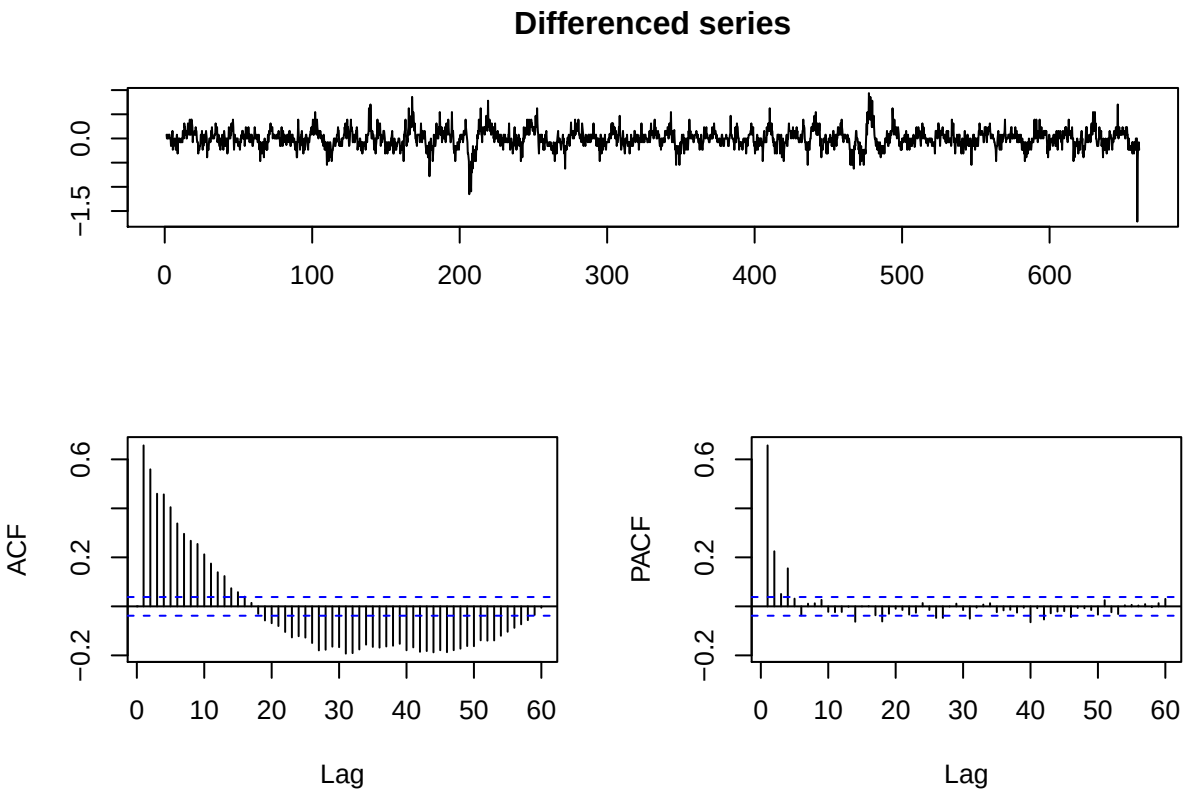
```
kpss.test(tempdryer)
```

```
## Warning in kpss.test(tempdryer): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tempdryer
## KPSS Level = 1.5115, Truncation lag parameter = 11, p-value = 0.01
```

The low p-value points to the presence of evidence in data in favour of integrating effects. The next step is to determine the *degree of differencing* required to eliminate these integrating effects. We can do this either manually or automatically using the `ndiffs` routine of the `forecast` package.

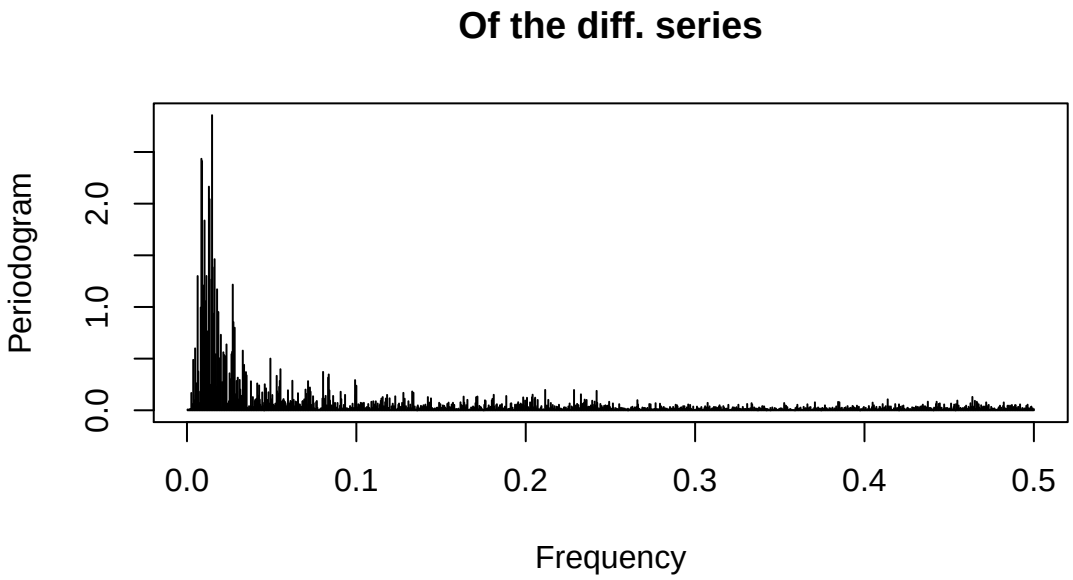
```
difftimes <- ndiffs(vk)
vkd <- diff(vk, difftimes)
tsdisplay(vkd, main = "Differenced series", lag.max = 60, points = F)
```



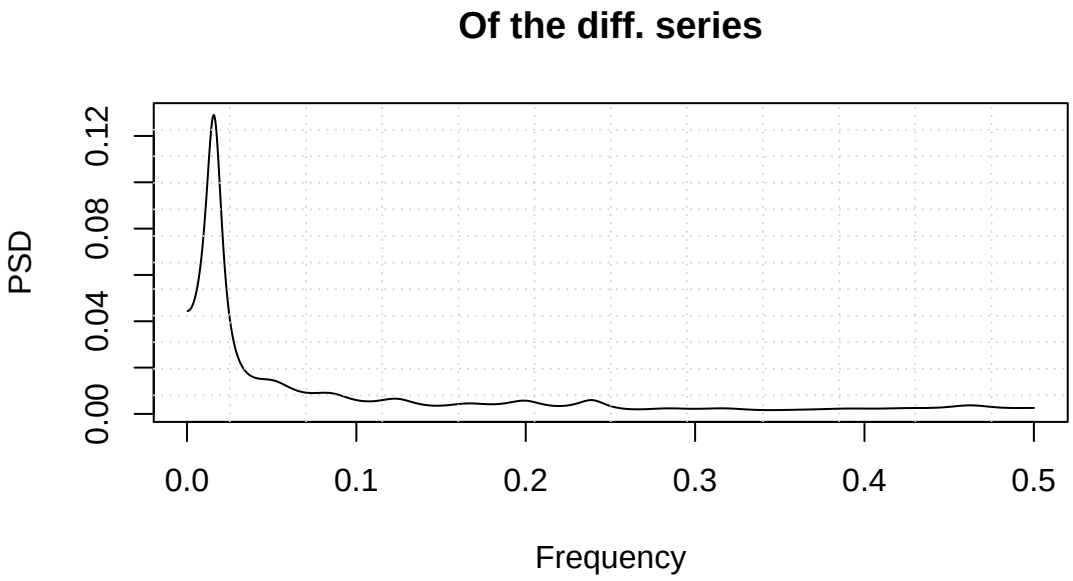
Comparing the ACFs of the differenced ($\nabla v[k]$ and the original series ($v[k]$), it is clear that a single degree of differencing has eliminated the integrating effect. Furthermore, the periodicities are more pronounced in the differenced series (to be on the safe side, it is a good idea to conduct the KPSS test on the differenced

series $\nabla v[k]$). These observations can also confirmed by examining the raw periodogram and parametrically estimated spectral densities of $\nabla v[k]$.

```
pgramvkd <- periodogram(vkd, plot = F)
psdvkd <- spec.ar(vkd, length(pgramvkd$freq), plot = F)
plot(pgramvkd$freq, pgramvkd$spec, main = "Of the diff. series", ylab = "Periodogram",
    xlab = "Frequency", lwd = 1, type = "h")
```




```
plot(pgramvkd$freq, psdvkd$spec, main = "Of the diff. series", ylab = "PSD",
    xlab = "Frequency", type = "l")
grid(12, 12)
```



The first plot is the raw periodogram (obtained by the usual periodogram expression), while the second plot is the parametric estimate of the spectral density, both of $\nabla v[k]$ (an AR(27) model was fit by the `spec.ar` routine for the parametric estimator). Observe that the parametric estimate clearly shows the presence of a periodicity that is not easily visible in the periodogram (due to the strong noisy other component).

Finally, it is of interest to note that the variance of the original series $v[k]$ is 4.3755849 and the differenced series $\nabla v[k]$ is 0.0336225, indicating that a significant fraction of the original variation has been captured by

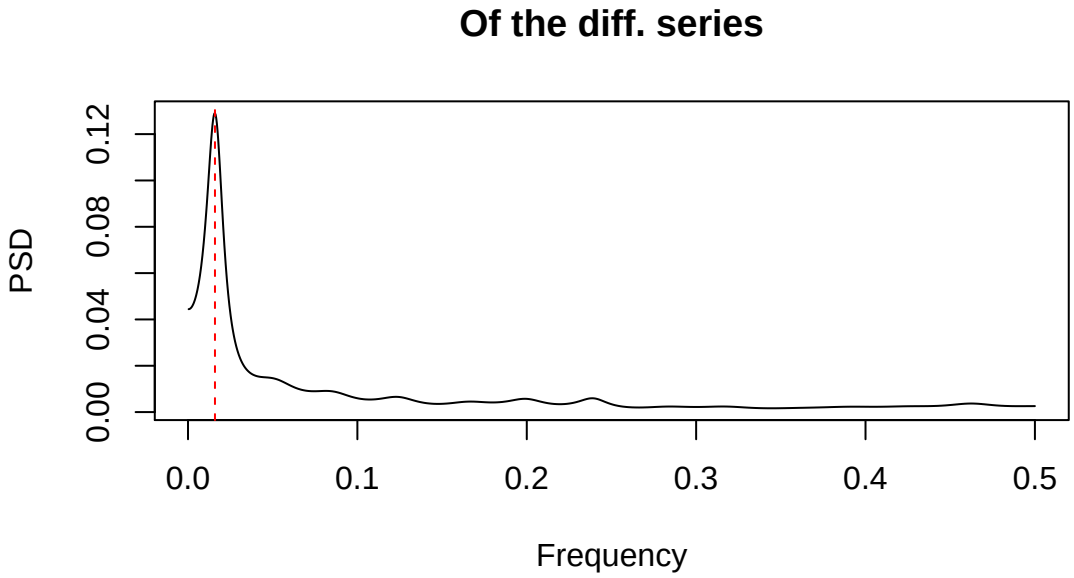
the integrator.

The peak in the PSD plot occurs at a frequency of 0.129 cycles/sample. The following R chunk helps in locating the peak (the dashed red line corresponds to the peak).

```
require(splus2R)
```

```
## Loading required package: splus2R
```

```
# Specify the span as per your requirement
psd_peaks <- peaks(psdvkd$spec, span = 51)
# Plot the PSD with the peak locations
plot(pgramvkd$freq, psdvkd$spec, main = "Of the diff. series", ylab = "PSD",
    xlab = "Frequency", type = "l")
# Mark only the first peak
abline(v = pgramvkd$freq[which(psd_peaks)[1]], col = "red", lty = "dashed")
```



At this juncture, we have three options for the differenced series:

1. Fitting a sinusoidal component of the appropriate frequency and building a time-series model for the remainder of the series.
   This option is ideal when *sharp peaks* are noticed in the periodogram (or spectral density). The present series does not have such a feature. Referring to the periodogram, the power is spread in a *band* around the identified peak frequency. Therefore, we do not pursue this option.
2. Fitting an ARMA model of appropriate order (that will hopefully model the peak in the PSD).
   ARMA models are capable of modelling random processes with spectral densities spread over broad-band or localized in a somewhat narrow band or containing a mix of periodicities in a concentrated band. The present series has the latter characteristics. In fact, we can term the series as **cyclic**. ARMA models are capable of modelling these series, provided the auto-regressive component is **at least of second-order.**
3. Fitting a seasonal ARIMA model of appropriate regular and seasonal orders with appropriate periodicity.
   SARIMA models are appropriate when the *spectral density exhibits peaks at frequencies that are integer multiples* of the fundamental frequency (since SARIMA models are characterized by such spectral densities). The PSD of the given series does not exhibit such a characteristic. Moreover, there is no reason to beleive that an industrial dryer can exhibit seasonality (as against cyclicity or a sinusodial trend). Therefore, we do not consider this as an option for this series (as a simple exercise the reader is encouraged to fit a SARIMA model and check if any of the model coefficients can be estimated reliably).

Finally, also observe that the series does not exhibit any deterministic trends.

With the above arguments in place, we now proceed to developing a suitable ARMA model for the differenced series, i.e., an ARIMA(P,1,M) for the original series.

Developing the time-series model

Before we proceed to developing the time-series model, it is useful to have a function that performs the routine diagnostics that one generates from an ARIMA model. There exists the `tsdiag` routine in the `stats` package, but it is useful to write a customized one so that we are fully aware of the proceedings. For this purpose, I have written a function named `mytsdiag.R`, the code for which is provided below.

```
mytsdiag <- function(modarima, Lmax = 30) {
    # Print summary
    summary(modarima)
    # Extract residuals and plot them
    err_mod <- modarima$residuals
    N = length(err_mod)
    # layout(matrix(c(1,2,3),3,1,byrow=TRUE),heights=c(1,1,1))
    par(mfrow = c(3, 1), mai = c(0.6, 0.7, 0.2, 0.2))
    plot(scale(err_mod), type = "l", ylab = "Std. resid.", xlab = "")
    # Compute ACF of residuals
    acf_err <- acf(err_mod, lag.max = Lmax, main = "", plot = F)
    lowsig = -1.96/sqrt(N)
    upsig = 1.96/sqrt(N)
    plot(acf_err$lag * tsp(err_mod)[3], acf_err$acf, type = "h", main = "",
        ylab = "ACF of Resid", xlab = "", ylim = c(1.2 * lowsig, 1.2 * upsig))
    abline(h = upsig, col = "red", lty = "dashed")
    abline(h = lowsig, col = "red", lty = "dashed")
    abline(h = 0, col = "black")
    # Compute BLP statistic
    blpval <- NULL
    npar <- sum(modarima$arma[1:4])
    Lval <- (npar + 1):Lmax
    for (L in Lval) {
        blpval <- c(blpval, Box.test(modarima$residuals, lag = L, fitdf = npar)$p.value)
    }
    # Plot BLP statistic
    plot(1:Lmax, c(rep(NA, npar), blpval), ylab = "p-values", xlab = "Lag",
        ylim = c(0, 1))
    abline(h = 0.05, col = "red", lty = "dashed")
}
```

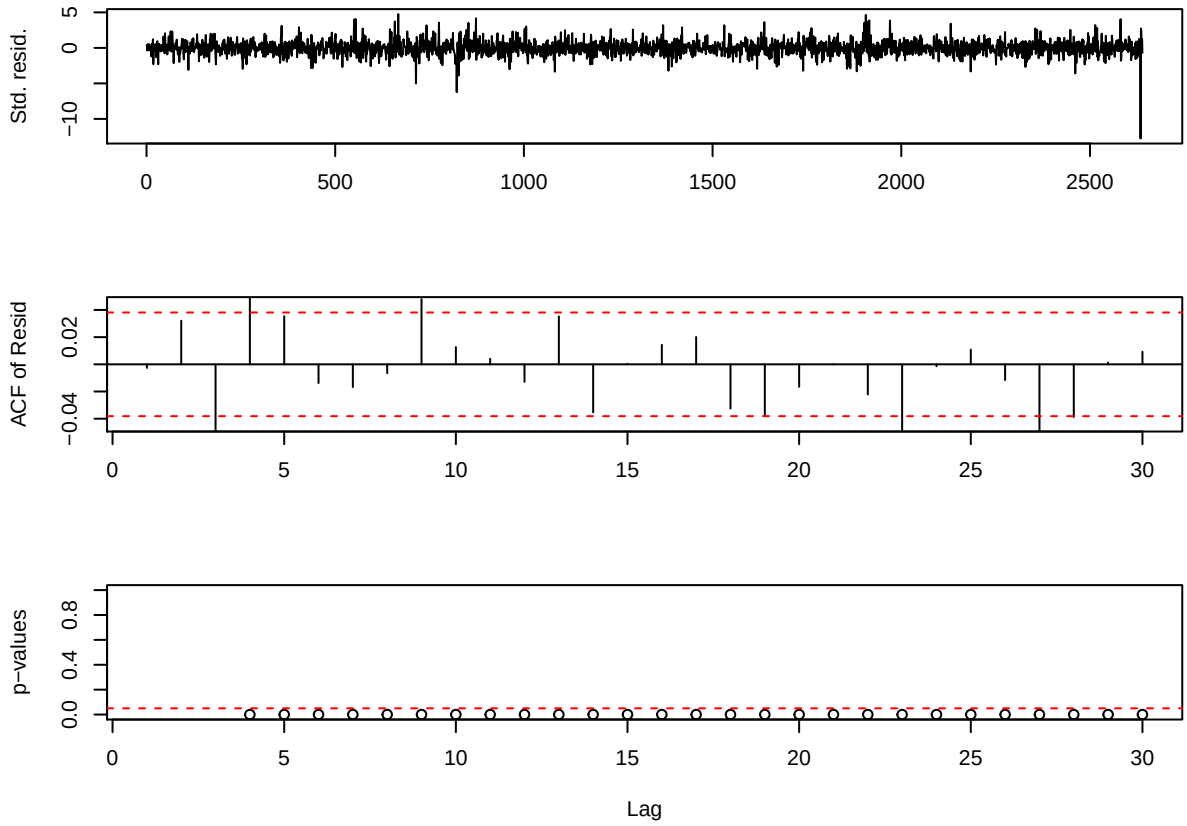
As it can be seen, the code prints the summary diagnostics (coefficients, standard errors, AIC, MSE, etc.), plots the *standardized* residuals, ACF of the residuals and the p-value from the BLP test. A "good" model should result in white residuals as confirmed by both ACF and BLP test (p-value greater than the significance level α). Note that the minimum lag L in the BLP test is $\dim\theta + 1$, where θ is the set of (ARMA and seasonal) parameters being estimated.

Keeping in mind the foregoing discussion, we begin with an ARMA(2,1) model (i.e., a minimum order of two for the AR component).

```
modarma21 <- stats::arima(vkd, order = c(2, 0, 1), include.mean = F)
mytsdiag(modarma21)
```

```
##
```

```
## Call:
## stats::arima(x = vkd, order = c(2, 0, 1), include.mean = F)
##
## Coefficients:
##          ar1      ar2      ma1
##       1.0341  -0.1267  -0.5520
## s.e.  0.0575   0.0446   0.0523
##
## sigma^2 estimated as 0.01789:  log likelihood = 1564.11,  aic = -3120.22
##
## Training set error measures:
##                       ME      RMSE        MAE MPE MAPE      MASE
## Training set -0.0004646859 0.1337522 0.09612515 NaN  Inf 0.8871943
##                     ACF1
## Training set -0.0026253
```

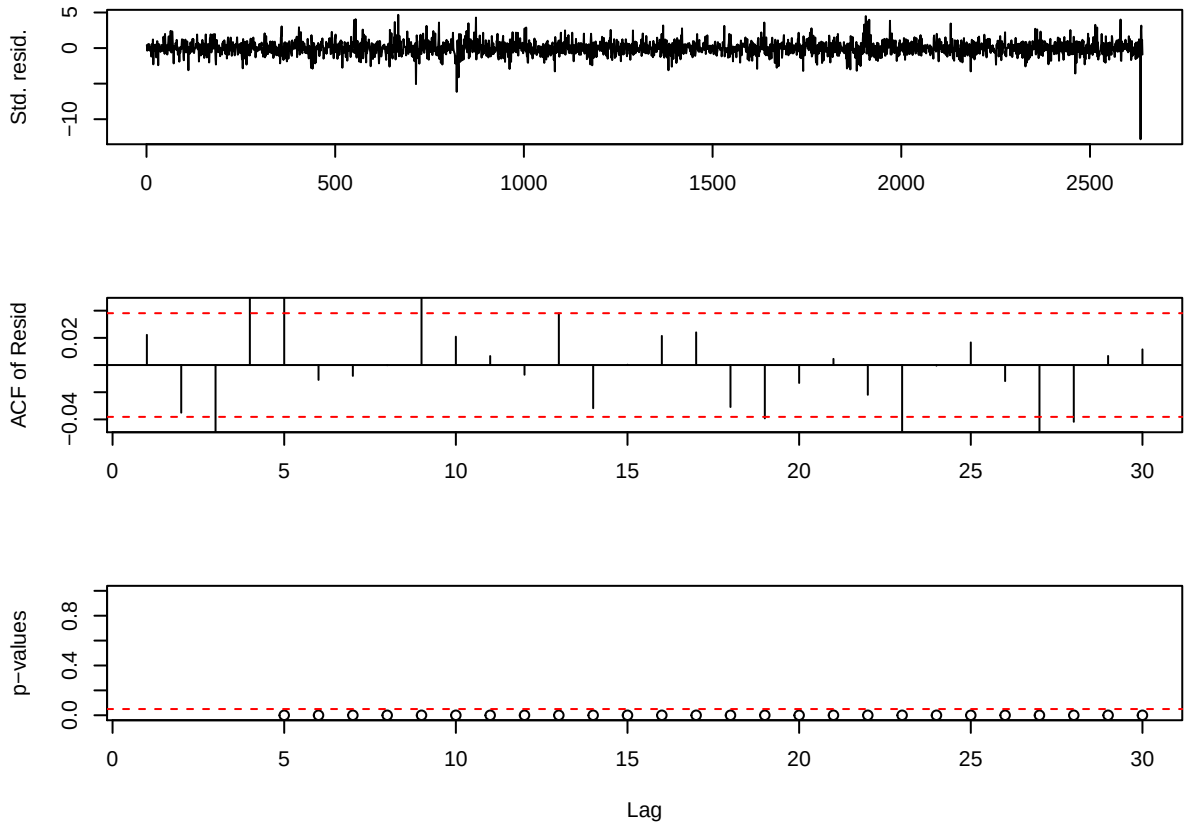


Clearly, the ARMA(2,1) model (on the differenced series) is not satisfactory. Increasing the MA order to 2 also does not improve the situation as seen below.

```
modarma22 <- stats::arima(vkd, order = c(2, 0, 2), include.mean = F)
mytsdiag(modarma22)
```

```
##
## Call:
## stats::arima(x = vkd, order = c(2, 0, 2), include.mean = F)
##
## Coefficients:
##          ar1     ar2     ma1      ma2
##       0.1022  0.6517  0.3612  -0.2740
```

```
## s.e.  0.1138  0.0998  0.1163   0.0569
##
## sigma^2 estimated as 0.01792:  log likelihood = 1561.72,  aic = -3113.44
##
## Training set error measures:
##                         ME      RMSE        MAE MPE MAPE      MASE
## Training set -0.000461324 0.1338737 0.09627324 NaN  Inf 0.8885611
##                    ACF1
## Training set 0.0222151
```



This is clearly not the run-of-the-mill series! At this stage we take the "brute force" route with AIC as our guide. Taking this route, we scan through all ARMA models of AR orders 2 to 5 and MA orders 1 to 5. The script for this exercise is given below. Note that the routine `auto.arima` from the `forecast` package is available, but this routine is **strongly not recommended*.

```
# Loop through orders
Pmin = 2
Mmin = 1
Pmax = 5
Mmax = 5
arnum = Pmax - Pmin + 1
manum = Mmax - Mmin + 1
modnum <- arnum * manum

modlist <- list()
aicmat <- matrix(NA, nrow = arnum, ncol = manum)
# Change the names of AIC matrix
rownames(aicmat) <- paste("AR", Pmin:Pmax, sep = "")
colnames(aicmat) <- paste("MA", Mmin:Mmax, sep = "")
```

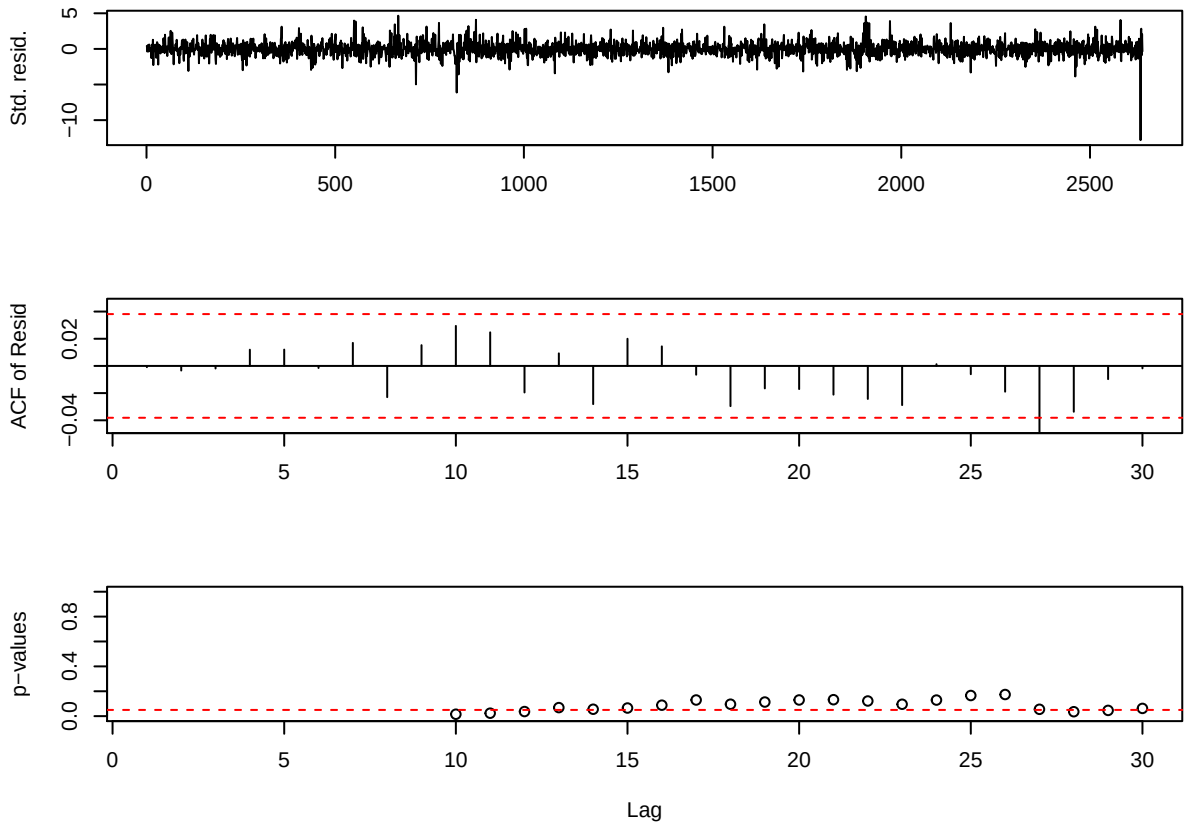
```
for (arord in (Pmin:Pmax)) {
    for (maord in (Mmin:Mmax)) {
        modarmaxx <- stats::arima(vkd, order = c(arord, 0, maord), include.mean = F)
        modstr <- paste("m", arord, maord, sep = "")
        modlist[[modstr]] <- modarmaxx
        aicmat[arord - Pmin + 1, maord - Mmin + 1] = modarmaxx$aic
    }
}

# Find the model with minimum AIC
minaic <- which(aicmat == min(aicmat), arr.ind = T)
modstrmin <- paste("m", paste(minaic, collapse = ""), sep = "")
print(modstrmin)
```

```
## [1] "m45"
```

```
# Retrieve that model
modbestaic <- modlist[[modstrmin]]
mytsdiag(modbestaic)
```

```
##
## Call:
## stats::arima(x = vkd, order = c(arord, 0, maord), include.mean = F)
##
## Coefficients:
##          ar1      ar2     ar3     ar4      ma1     ma2     ma3      ma4
##       0.4974  -0.5171  0.3314  0.4023  -0.0103  0.6933  -0.032  -0.1684
## s.e.  0.1711   0.1667  0.1493  0.1176   0.1708  0.0911   0.142   0.0681
##           ma5
##       -0.0738
## s.e.   0.0289
##
## sigma^2 estimated as 0.01755:  log likelihood = 1589.56,  aic = -3159.12
##
## Training set error measures:
##                        ME      RMSE        MAE MPE MAPE      MASE
## Training set -0.000440249 0.1324649 0.09573369 NaN  Inf 0.8835813
##                       ACF1
## Training set -0.0009206734
```



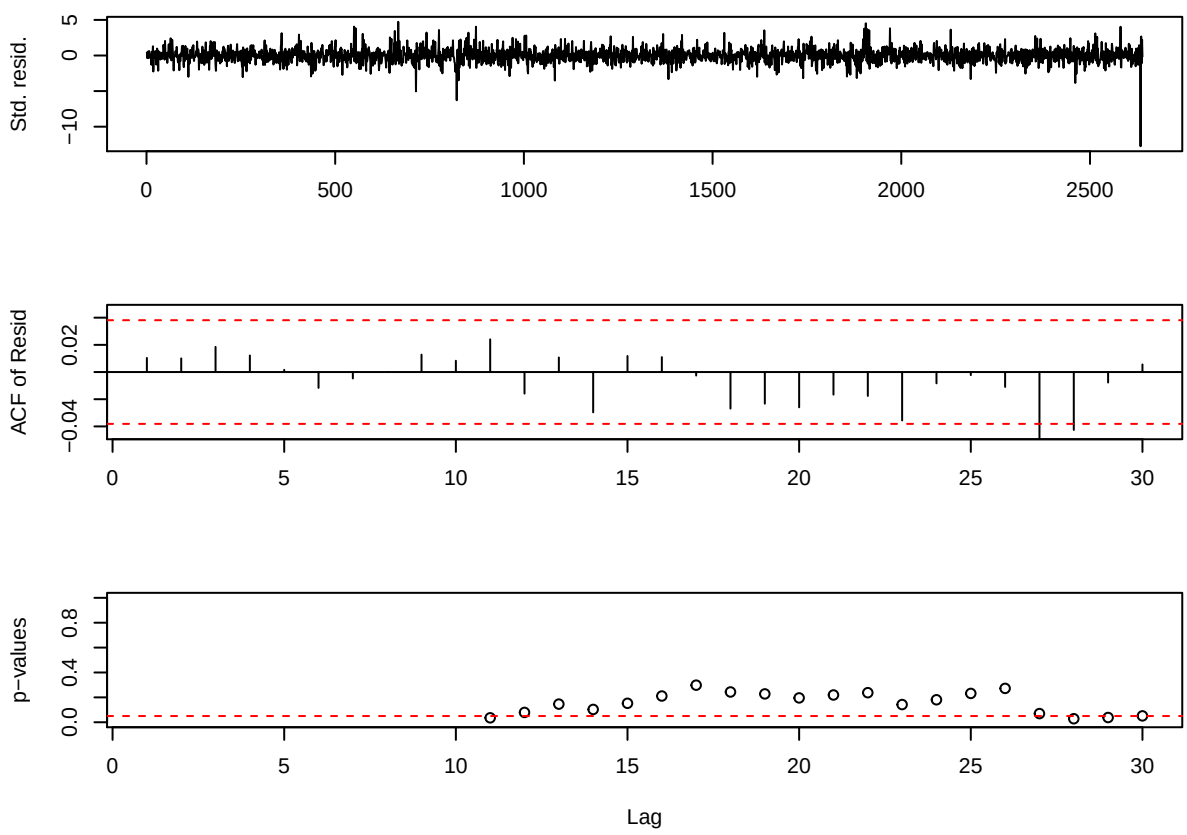
AIC picks the best model as ARMA(4,5). The resulting model is somewhat satisfactory (not in an ideal sense) since the BLP test yields low p-values at high lags, indicating some unexplained effects in the residuals. Keeping aside this aspect, we turn our attention to the standard errors in the coefficient estimates. Clearly some of these coefficient estimates are not significant. Thus, we observe that AIC cannot pay attention to all aspects of modelling. Interestingly, the estimated model suggests that some of the lower-order and intermediate coefficients are insignificant.

With the model guided by AIC as an initial guess we proceed to developing a **constrained** ARMA model, the *constraints* being that some of the coefficients in the model be zero-valued. After a careful search around this model, it turns out that the model that is almost satisfactory in all respects is a **constrained** ARMA(5,5) model. The script below estimates this model and generates the necessary diagnostics.

```
# Estimate constrained model
modarma55c <- stats::arima(vkd, order = c(5, 0, 5), include.mean = F, fixed = c(0,
    NA, NA, NA, NA, NA, NA, 0, 0, NA), transform.pars = F)
# Diagnose the resulting model
mytsdiag(modarma55c)
```

```
##
## Call:
## stats::arima(x = vkd, order = c(5, 0, 5), include.mean = F, transform.pars = F,
##     fixed = c(0, NA, NA, NA, NA, NA, NA, 0, 0, NA))
##
## Coefficients:
##       ar1      ar2     ar3     ar4     ar5     ma1     ma2  ma3  ma4
##         0  -0.5386  0.4814  0.2759  0.3672  0.4772  0.9386    0    0
## s.e.    0   0.0776  0.0364  0.0263  0.0445  0.0210  0.0719    0    0
##           ma5
##       -0.2925
```

```
## s.e.   0.0475
##
## sigma^2 estimated as 0.01756:  log likelihood = 1588.31,  aic = -3160.63
##
## Training set error measures:
##                         ME     RMSE        MAE MPE MAPE      MASE
## Training set -0.0004475856 0.132528 0.09571484 NaN  Inf 0.8834073
##                     ACF1
## Training set 0.01038732
```



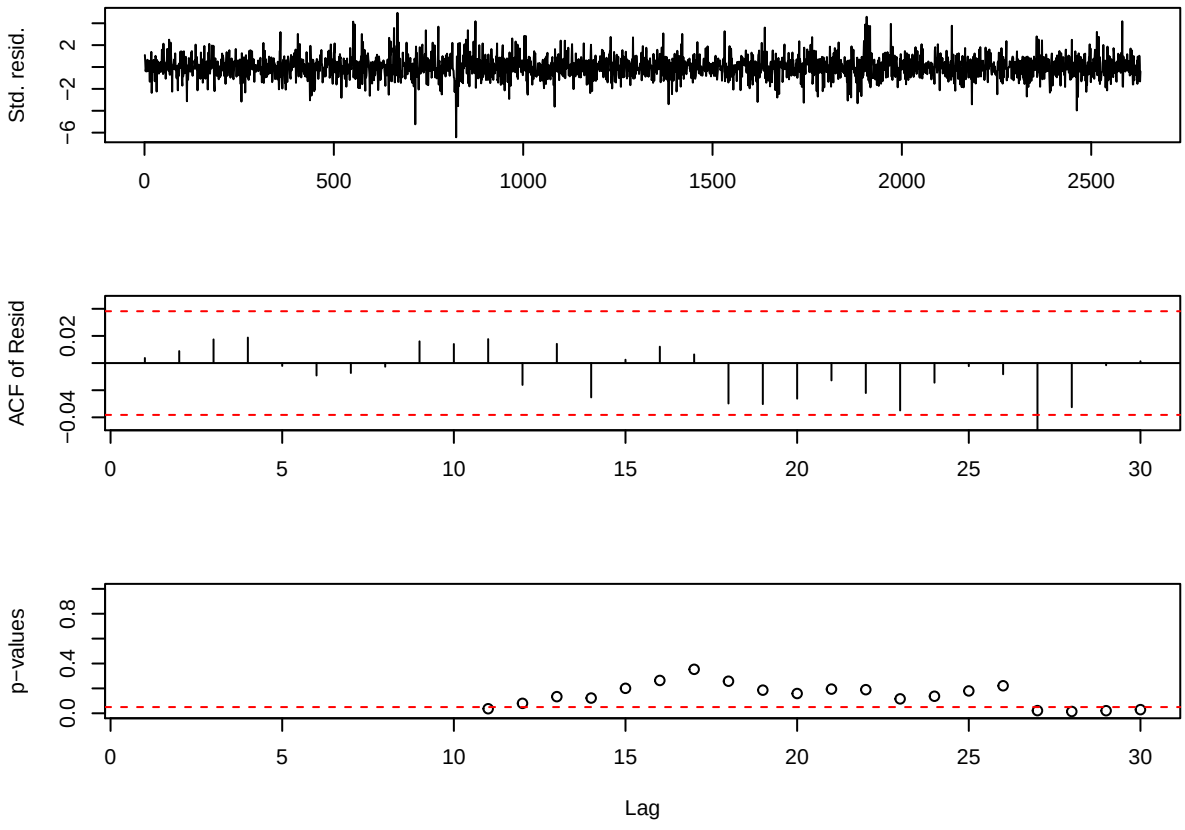

```
# Compute confidence intervals for parameters
confint(modarma55c)
```

```
##          2.5 %     97.5 %
## ar1         NA         NA
## ar2 -0.6906556 -0.3865974
## ar3  0.4101058  0.5526077
## ar4  0.2243644  0.3274891
## ar5  0.2800810  0.4543900
## ma1  0.4360602  0.5183252
## ma2  0.7977434  1.0794285
## ma3         NA         NA
## ma4         NA         NA
## ma5 -0.3856240 -0.1994287
```

Barring the low p-values at higher lags in the BLP test, the model is satisfactory in all respects. Note that this is the best model one could build in the ARIMA framework for this time-series. Quite a challenging one! Putting together everything that we have had, we now proceed to building a **constrained** ARIMA(5,1,5) model by leaving out the last 10 observations for forecasting.

```r
# Partitioning the data and performing cross-validation
modarma55ct <- stats::arima(vk[1:2630], order = c(5, 1, 5), include.mean = F,
    fixed = c(0, NA, NA, NA, NA, NA, NA, 0, 0, NA), transform.pars = F)
# Diagnosing the model
mytsdiag(modarma55ct)
```

```
##
## Call:
## stats::arima(x = vk[1:2630], order = c(5, 1, 5), include.mean = F, transform.pars = F,
##     fixed = c(0, NA, NA, NA, NA, NA, NA, 0, 0, NA))
##
## Coefficients:
##       ar1      ar2     ar3     ar4     ar5     ma1     ma2  ma3  ma4
##         0  -0.4662  0.4666  0.2760  0.3155  0.4992  0.8907    0    0
## s.e.    0   0.0977  0.0526  0.0241  0.0427  0.0197  0.0966    0    0
##           ma5
##       -0.2517
## s.e.   0.0458
##
## sigma^2 estimated as 0.01637:  log likelihood = 1674.56,  aic = -3333.11
##
## Training set error measures:
##                          ME      RMSE        MAE         MPE       MAPE
## Training set -0.0001702957 0.1279623 0.09499019 1.56619e-05 0.06806099
##                    MASE        ACF1
## Training set 0.7282228 0.003738536
```



```
# Predict the next 10 observations
dev.new()
tsattr <- tsp(vk)
vk2 <- ts(as.vector(vk), start = 1, end = length(vk), frequency = 1)
hlen = length(vk) - length(modarma55ct$residuals)
if (hlen >= 0) {
    vkpred <- forecast(modarma55ct, h = hlen)
    # Plot the one-step ahead forecasted series
    plot(vkpred)
} else {
    print("No data remaining to compare forecasts")
}
# Plot the original series
lines(vk2, col = "red")
```

Discussion

From the plot it is clear that the **10-step** ahead forecast is not that encouraging. This is to be expected since the series does not seem to fit within the linear stationary framework. In fact, the series is heteroskedastic (for the interested, the `stationarity` routine in the `fractal` package runs a test of heteroskedasticity). Therefore, we may have to turn to GARCH type models or a non-linear time-series model if we wish to have better h-step ahead forecasts. Nevertheless, the one- and two-step ahead forecasts of the developed model are fairly satisfactory.

The final model that we have for the series is an ARIMA(5,1,5) model

$$\nabla v[k] = \frac{1 + c_1 q^{-1} + c_2 q^{-2} + c_5 q^{-5}}{1 + d_2 q^{-2} + d_3 q^{-3} + d_4 q^{-4} + d_5 q^{-5}} e[k], \qquad \sigma_e^2 = 0.0164$$